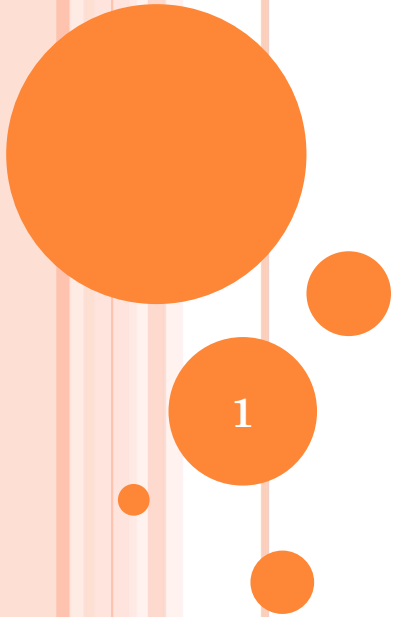
Helper class in AWT



Graphics class methods

- In GUI applications, we can use Graphics class of java.awt package to create various graphics like lines, rectangles, circles, polygons etc.
- A graphics context provides the capabilities of drawing on the screen.
- The graphics context maintains states such as the color and font used in drawing, as well as interacting with the underlying operating system to perform the drawing.
- Let's look at some of the methods available in the Graphics class:

1. Drawing Lines

- The drawLine( ) method draws a line between two points specified by a pair of coordinates.
- The syntax is,

  void drawLine(int x1, int y1, int x2, int y2)
- Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.

2. Drawing Rectangle

void drawRect(int top, int left, int width, int height)

- This method draws the outline of a rectangle specified by a coordinate pair, a width, and a height.
- Here upper-left corner of the rectangle is specified at top and left parameters.

void fillRect(int top, int left, int width, int height)

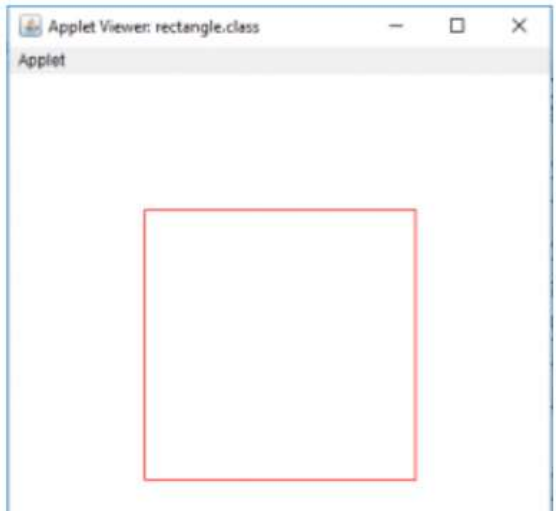
- is used to draw fill rectangle with the default color and specified width and height.

void drawRoundRect(int top, int left, int width, int height, int arcWidth, int arcHeight)

- Draws an outlined round-cornered rectangle using current color.
- The extra parameters arcWidth and arcHeight indicates how much of corners will be rounded.

void fillRoundRect(int top, int left, int width, int height, int arcWidth, int arcHeight)

- is used to draw fill rounded rectangle with the default color and specified width and height



3. Drawing ellipse and circles

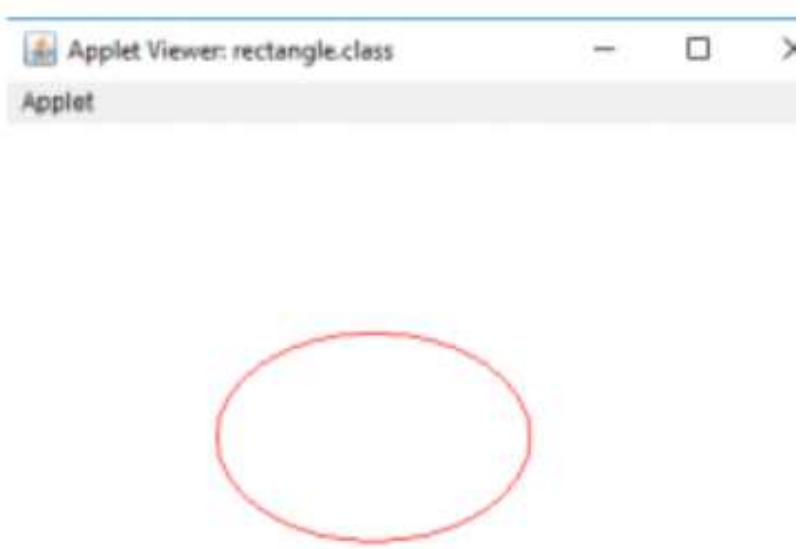
- drawOval( ) method is used to draw a circle or ellipse.

void drawOval(int top, int left, int width, int height)

- is used to draw oval with the specified width and height.
- The first 2 arguments represents the top left corner and the 2 arguments represents the width and height of the oval.
- If the width and height are the same the oval becomes the circle.

void fillOval(int top, int left, int width, int height)

- is used to draw fill oval with the default color and specified width and height.



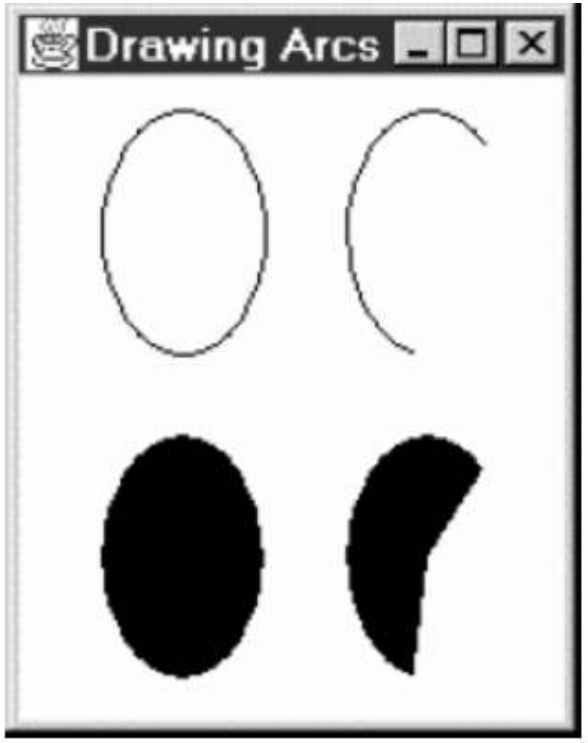
4. Drawing arcs

- An arc is part of an oval or circle.
- drawArc( ) method is used to draw arcs.
- This method have 6 arguments.
- The first four arguments are same arguments in drawOval( ) method.
- The last two arguments represents the starting angle of the arc and the number of degrees around the arc.
- This is known as sweep angle.

- The general form is,

void drawArc ((**int top, int left, int width, int height, int startAngle, int arcAngle)**

void fillArc(int top, int left, int width, int height, int startAngle, int arcAngle)



5. Drawing polygons

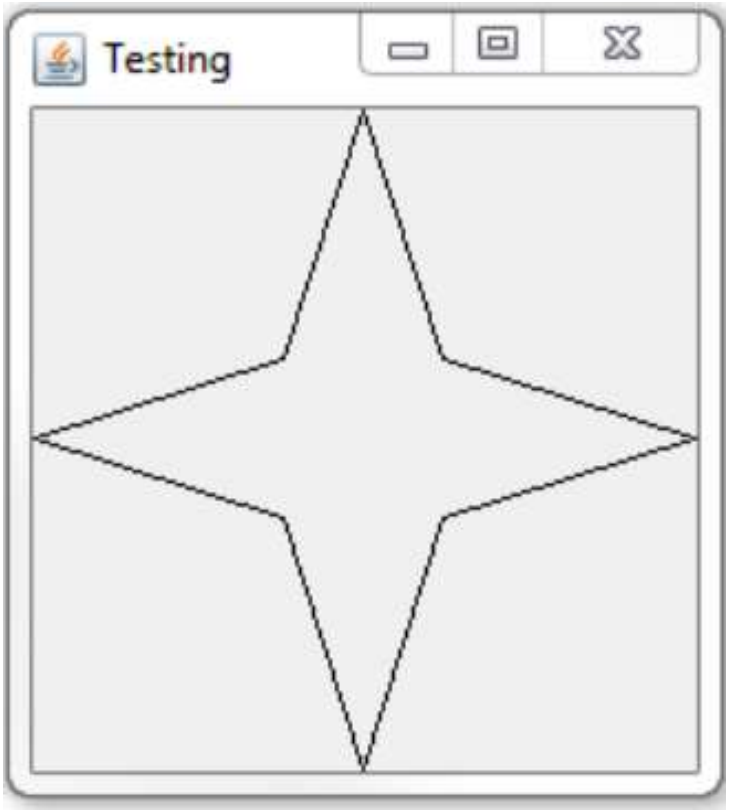
- Polygons are the shapes with many sides.
- A polygon may be considered as a set of lines connected together.
- The end of one line is the beginning of the next line.
- That means we can draw a polygon with n sides using the drawLine( ) method n times
- We can draw the polygons more conveniently using drawPolygon( ) method.

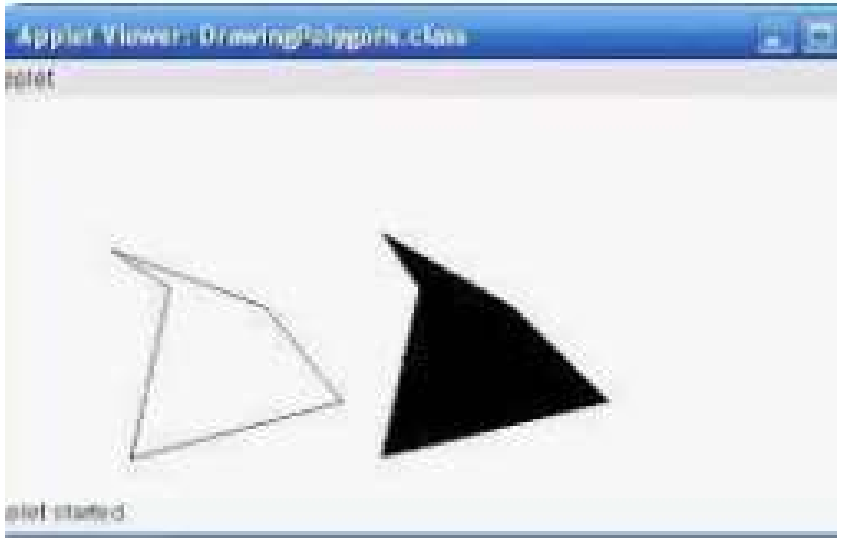
- The general form is,

void drawPolygon(int x[ ], int y[ ], int nPoints)

void fillPolygon(int x[ ], int y[ ], int nPoints)

- ✓ Here the first argument represents an array of integers containing X coordinates
- ✓ The second argument represents an array of integers containing Y coordinates
- ✓ The third argument represents the size of the coordinates.
- ✓ Here the 2 arrays should of the same size.


Testing


Applet Viewer: Drawing Polygons.class
Applet
Applet started

WORKING WITH COLORS

- The Color class is a part of Java Abstract Window Toolkit(AWT) package.
- The Color class creates color by using the given RGB color system.
- RGB color specifies red, green and blue.
- The value for individual components RGB ranges from 0 to 255 or 0.0 to 0.1.
- The value determines the opacity of the color, where 0 or 0.0 stands fully transparent and 255 or 1.0 stands opaque.

Constructors of Color Class

Color(float r, float g, float b)

- creates a opaque color with specified RGB components(values are in range 0.0 – 0.1)

Color(int r, int g, int b)

- Creates a opaque color with specified RGB components(values are in range 0 – 255)

- For eg,

  Color c1=new Color(255,100,100);

- This statement creates a specific color with R, G and B values are 255, 100,100
- We can also create the static colors.

  Color c = Color.yellow;

void setColor(Color c)

- **This method** is used to set the graphics current color to the specified color in the argument c.
- This method is in Graphics class

- To set the background color of an applets window, use setBackground().
- To set the foreground color (the color in which text is shown, for example), use setForeground().
- These methods are defined by Component class in java.awt package, and they have the following general forms:
  - void setBackground(Color newColor)
  - void setForeground(Color newColor)

Here, newColor specifies the new color.

- For example, this sets the background color to green and the text color to red:

```
setBackground(Color.green);
setForeground(Color.red);
```

- We can obtain the current settings for the background and foreground colors by calling getBackground() and getForeground( ), respectively.
- They are also defined by Component and are shown here:
  - Color getBackground( )
  - Color getForeground( )

Font class in Java

- The java.awt package contains a Font class which provides the mechanism for setting the attributes of the font used.
- To select a new font, we must first construct a Font object that describes that font.
- A Font constructor has this general form:

  Font(String fontName, int fontStyle, int pointSize)

  - Here fontName refers to the name of the font, fontStyle refers to the style of the font, which is a constant defined in Font class which takes values like BOLD,ITALIC,PLAIN and pointSize refers to the size of the font.

- Eg:

  Font f = new Font("Helvetica", Font.BOLD , 22)
- This statement will create a bold Helvetica, 22-point font.
- After creating the font the setFont() method selects the font into the current graphics context. This method is defined as follows:

  void setFont(Font font)

- Eg:

  g.setFont(f);

- where g is the Graphics object
- Now drawString() method can be used to draw some text using the font you've created, sized up, and selected.

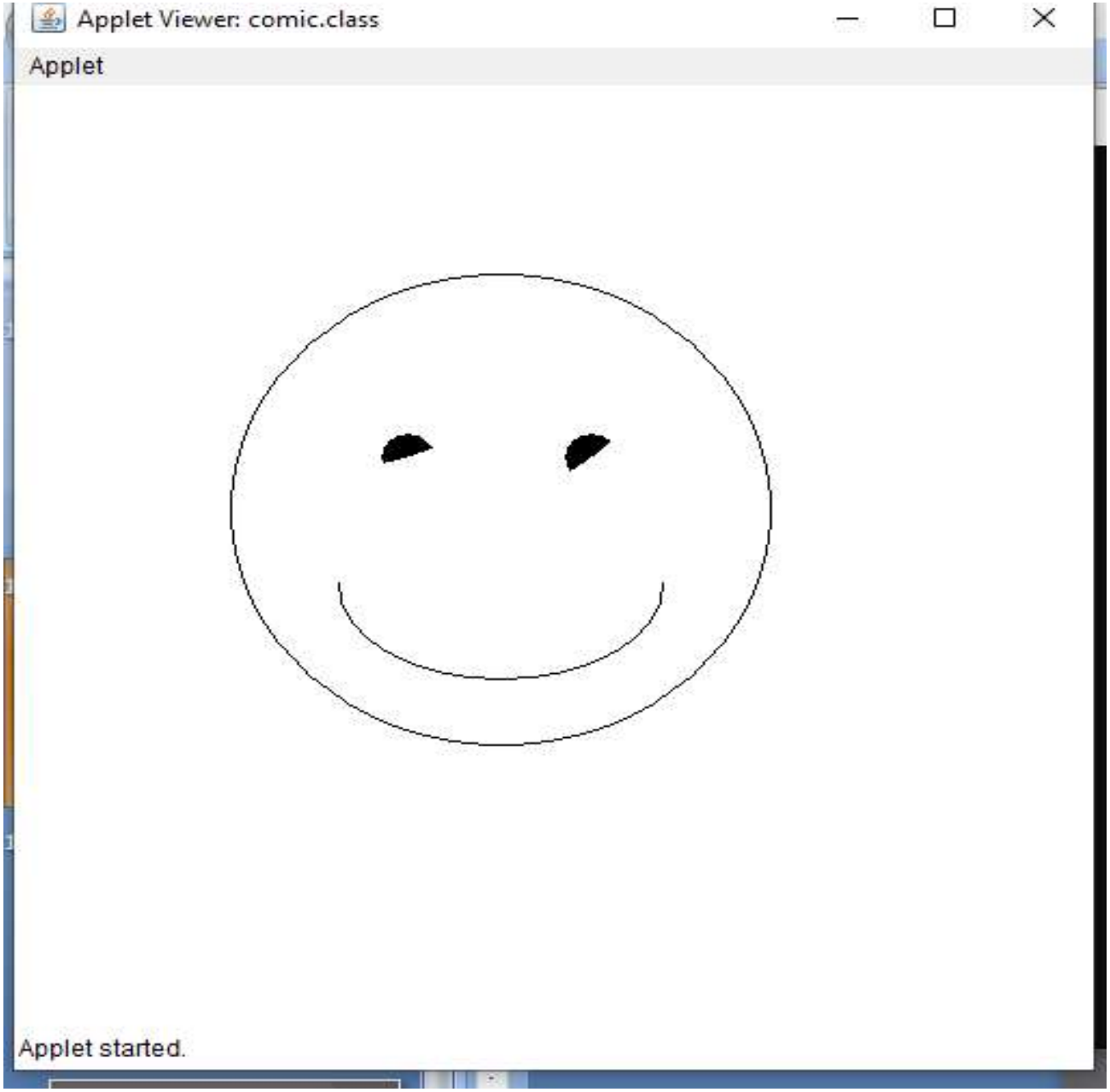
```
Font f = new Font("TimesRoman", Font.PLAIN, 34);
g.setFont(f);
g.drawString("This is a big font.", 10, 50);
```

Paint() method

- The class Component defines methods called paint(), repaint() and update().
- The way these methods interact can sometimes create strange results.
- First, consider what the methods are intended for:
- paint(): paint() is where you place code for drawing, writing etc. Initial placement of buttons and other widgets should not be done in paint(). Whenever a rendering is thought to be "damaged" (e.g, partially covered by another window), paint() is called to re-draw the scene. Note: AWT maintains information about the size of the "drawn" area, so that if a portion of the window outside this area is covered and un-covered, paint() may not be called. Thus, if buttons are created in paint, new buttons will be created each time paint() is called. This is why you shouldn't create buttons in paint(). Note: paint() is given a Graphics context as a parameter.

- update(): update is called when the window is re-sized. The default implementation of update(): first clears the background; then calls paint().
- repaint(): The repaint() is intended to allow various methods to call for a re-rendering of the component. No graphics context is needed for repaint(). A call to repaint() calls update().
- Here's an example of a weird effect:

1. Shapes.java
2. **colorexample.java**
3. **comicface.java**


Applet Viewer: comic.class
Applet
Applet started.

EXAMPLE PROGRAM DRAWING POLYGON

```
public void paint(Graphics g)
 {
   int x[] = { 70, 150, 190, 80, 100 };
   int y[] = { 80, 110, 160, 190, 100 };
   g.drawPolygon (x, y, 5);

   int x1[] = { 210, 280, 330, 210, 230 };
   int y1[] = { 70, 110, 160, 190, 100 };
   g.fillPolygon (x1, y1, 5);
 }
```